APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DETERMINING A NODE PATH THROUGH A NODE GRAPH

APPLICANT: CARL S. MARSHALL AND ADAM T. LAKE

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL935340392US

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

January 4, 2002
Date of Deposit

Signature

Gabe Lewis
Typed or Printed Name of Person Signing Certificate

DETERMINING A NODE PATH THROUGH A NODE GRAPH

TECHNICAL FIELD

This invention relates to determining a node path through a node graph relating to a three-dimensional (3D) mesh.

BACKGROUND

A 3D mesh is made up of one or more polygons that define a surface, such as a terrain. The number of polygons in the 3D mesh may be increased, resulting in increased resolution, or decreased, resulting in decreased resolution. Increasing the number of polygons in the 3D mesh increases the resolution of the surface by making the surface more detailed and decreasing the number of polygons in the 3D mesh decreases the resolution of the surface by making the surface less detailed.

Decreasing the resolution of the surface can increase rendering performance, particularly on low-end hardware. That is, since there are fewer polygons to process, the 3D mesh can be manipulated using a less powerful graphics processor and/or using fewer processor cycles. This could also relate to a node graph defined by a 3D mesh.

DESCRIPTION OF DRAWINGS

Fig. 1 is view of a 3D mesh.

Fig. 2 is a view of a surface terrain represented by the 3D mesh.

Fig. 3 is a view of a polygon in the 3D mesh.

Fig. 4 is a flowchart showing a process for determining a node path through the 3D mesh.

Fig. 5 is a view of several polygons in the 3D mesh.

Fig. 6 is a view of a reduced-resolution (fewer polygons) version of the 3D mesh.

Fig. 7 is a view of a computer system on which the process of Fig. 4 may be implemented.

Like reference numerals in different figures indicate like elements.

DETAILED DESCRIPTION

A 3D mesh may be used to represent a node graph. In this context, a node graph is a collection of nodes that define features of an environment, such as a terrain. The nodes may be positioned in 3D space to define the length, width and height of environmental features. An animated model traverses the node graph by "interpolating" through the nodes, which means that they are moving in between the nodes. Also, the

nodes can be control points for a spline which the model uses as a node path. In this regard, a route that the model takes through the node graph is referred to as the node path.

Referring to Fig. 1, a 3D mesh 10, which may be used to represent a node graph of a terrain, is comprised of interconnecting polygons 12. Polygons 12 are triangles in this embodiment; however, other types of polygons, such as quadrilaterals, may be used instead of triangles.

3D mesh 10 defines a polygonal surface 14 (Fig. 2), here mountainous terrain, that can be traversed by a 3D model, such as a video game character. The polygonal surface and/or a corresponding node graph can be created via a parametric surface. The 3D data for surface 14 defines interconnecting polygons 12 that make up 3D mesh 10. A node graph for surface 14 is defined using polygons 12. That is, polygons in 3D mesh 10 are defined to be nodes within the node graph. Each polygon may be assigned a node or the nodes may be dispersed throughout 3D mesh 10 on non-adjacent polygons.

A node graph may contain one or more blocking nodes. A blocking node defines a point through which a 3D model cannot pass while traversing the node graph (via a node path). Data is associated with each blocking node, which indicates that a 3D model along the node path cannot pass through the blocking

node. The 3D model thus must go around the blocking node, rather than through it. Examples of objects defined by blocking nodes can include a tree, a wall, a building, a mountain, or any other non-permeable objects.

Referring to Fig. 3, the 3D data for a polygon 12 in 3D mesh 10 defines coordinates for three vertices 16a, 16b and 16c positioned in Cartesian XYZ (or other) 3D space. These vertices define a face 18 for polygon 12. The 3D data also defines a unit normal vector 20a, 20b and 20c to each vertex 16a, 16b and 16c and a unit normal vector 18a to face 18. The unit normal vectors for face 18 and vertices 16a, 16b and 16c have a magnitude of one and are orthogonal (i.e., normal) to face 18. These vectors are used to determine the shading of polygon 12 from a virtual light source.

Referring to Figs. 1 and 2, a node path 22 is also defined through 3D mesh 10. As noted, a node path is a route that is defined through a node graph. In the example of Fig. 1, node path 22 is a trail through surface 24.

Generally, a node path contains a starting node and a destination node. The starting node is the start point of the node path and may be associated with a node (in this case, a polygon of 3D mesh 10) at the beginning of the node path. The start and end of the node path are defined by the intended

movement of a 3D model along the node path from an initial position to a destination. In the example shown in Fig. 1, polygon 24 contains the start node.

The destination node is the end point of the node path and may be associated with a node (in this case, a polygon of 3D mesh 10) at the end of the node path. In the example shown in Fig. 1, polygon 26 contains the destination node.

Node path 22 may be defined by the data that makes up the node graph associated with 3D mesh 10. For example, the nodes of the node graph, and thus node path 22, may be located at centers of polygons on the path or at vertices of polygons on the path. Alternatively, the nodes of the node graph may be defined by data that is separate from 3D mesh 10. In this case, the node graph is superimposed over the appropriate portion of 3D mesh 10. Node graphs that are separate from the 3D mesh may contain associations to vertices on the mesh. Therefore, changing the 3D mesh also results in corresponding changes in the node graph (and thus, the node path). Consequently, node graphs that are defined by data that is separate from the 3D mesh can be processed similarly to node graphs that are defined by data associated with the 3D mesh.

Referring to Fig. 4, a process 28 is shown for determining a node path through a node graph, such as 3D mesh

10. Process 28 however, is not limited to use with a node graph defined by a 3D mesh, but rather can be used with any type of node graph defined by 3D or two-dimensional (2D) data.

In this embodiment, process 28 contains two stages: a pre-processing stage 30 and a run-time stage 32. Pre-processing stage 30 can be performed only once for a 3D animation sequence having multiple frames to be processed. If desired, pre-processing stage 30 can be performed several times randomly or at pre-specified time intervals. Run-time stage 32 is performed for each frame of an animation sequence.

Process 28 will be described with respect to 3D mesh 10 (Fig. 1) and surface 14 (Fig. 2). It is assumed that a node graph, and thus node path 22, through 3D mesh 10/surface 14 is defined by the 3D data that makes up 3D mesh 10.

In pre-processing stage 30, process 28 loads (401) 3D data for an animation sequence that includes 3D mesh 10. In this example, 3D mesh 10 is a single frame of the animation sequence. The 3D data includes the polygon structures shown in Fig. 1, along with data defining the node graph on 3D mesh 10. The 3D data may be loaded from memory into, e.g., a computer processing unit (CPU) that runs process 28.

Process 28 loads (402) update records into the CPU. The update records specify the number and locations of polygons to

remove, combine, or divide when adjusting the node graph through 3D mesh 10. As described below, the node graph is adjusted by changing the number of polygons in 3D mesh 10 using the update records.

Process 28 obtains (403) a metric that affects the way that process 28 operates. The metric may be obtained in any manner. For example, the metric may be retrieved from memory, downloaded from a network, or received via a graphical user interface (GUI) (not shown). In this embodiment, the metric relates to the performance of a platform (e.g., a CPU, graphics processor, operating system, or the like) that is running process 28. The metric may specify a frame rate for the animation sequence that contains the node graph, in this case that contains 3D mesh 10. For example, the metric may require the platform to maintain a frame rate of thirty frames-per-second during the animation sequence.

Process 28 modifies (404) 3D mesh 10, and thus the node graph, in accordance with the metric. Process 28 modifies (404) the 3D mesh by changing the number of polygons that make up 3D mesh 10. The number of polygons may be increased or decreased at this point. To increase the number of polygons, a standard subdivision technique may be used. This would be done, e.g., for high-speed processors that are capable of

handling large amounts of data. For the sake of illustration, however, it is assumed here that process 28 starts with a highest resolution 3D mesh and, therefore, the number of polygons is reduced, thereby reducing the resolution of the node graph. A reduction in the number of polygons in 3D mesh 10 enables the CPU or graphics processor running process 28 to comply with the metric (minimum frame rate).

One technique that may be used to reduce the resolution of 3D mesh 10 is the multi-resolution mesh (MRM) technique. This technique involves removing vertices of polygons defined by the update records, particularly vertices that are interior to a 3D mesh, and then connecting unconnected vertices to form new, larger polygons. By way of example, as shown in Fig. 5, edge 30 of polygon 32 is interior to 3D mesh 41. Consequently, its removal will not have a dramatic effect either way on the resolution of the 3D mesh. Accordingly, edge 30 can be removed, along, e.g., with edges 34 and 36, by removing their respective vertices and combining the smaller polygons and produce a larger polygon 38.

Process 28 performs (405) a path finding process on the node graph defined by modified 3D mesh 10. The path finding process determines the path 22 a 3D model should take through the node graph defined by modified 3D mesh 10 to go from a

predetermined starting point to a predetermined destination. Typically, the shortest route is determined; however, this is not a requirement. Examples of standard path finding processes that may be used include the A* process, the Dijkstra process, the depth-first process, and the breadth-first process. Typically, path finding processes begin at a "start" node, examine nodes around the start node, and determine which node has the least cost in terms of distance keeping the destination node in mind. The path finding process then advances to that node and repeats itself until it reaches the destination node.

Process 28 determines (406) if there is a change in the performance of the platform during the animation sequence. A change in the performance of the platform may result from any number of occurrences. For example, other computer programs running at the same time as the animation sequence may drain the platform's resources, resulting in a change in the platform's performance. Generally speaking, a change in performance refers to deterioration in performance; however, an increase in performance may also be detected.

If process 28 detects a change in performance of the platform, process 28 adjusts (407) the node graph by adjusting 3D mesh 10 in accordance with the change in performance, e.g.,

to compensate for the change in performance. For example, the performance of the platform may deteriorate such that the platform can only run twenty frames-per-second of animation. In this case, process 28 may reduce the number of polygons in the 3D mesh, and thereby adjust the node graph.

Because there are fewer polygons to process, the platform will be able to increase the number of frames that it can process per second. Conversely, if the performance of the platform increases, process 28 may increase the number of polygons in the 3D mesh, thereby providing enhanced resolution for higher-powered machines.

In this example, to adjust the node graph, process 28 generates a new, lower-resolution version of 3D mesh 10. An example of a new version of 3D mesh 10 is shown in Fig. 6 and is labeled 40 to avoid confusion with the version shown in Fig. 1. 3D mesh 40 includes an adjusted node graph, and thus an adjusted node path 42. Adjusted node path 42 (Fig. 6) differs from node path 22 (Fig. 1) in that adjusted node path 42 is defined by fewer polygons than node path 22.

Process 28 determines (408) if one or more predetermined types of nodes has been removed by adjusting (407) the node graph. In this context, such nodes include a start node, a destination node, and/or a blocking node. If one (or more) of

these nodes has been removed from the 3D mesh, process 28 re-locates (409) the node on the 3D mesh and performs (410) the path finding process on the node graph with the re-located node. What is meant by "re-locate" in this context is to re-define the location of the predetermined node, which may, or may not, mean moving from a current path position.

Process 28 may re-locate (409) the node as follows. Process 28 may obtain a position on the adjusted 3D mesh 40 that corresponds to the node that was deleted, e.g., the original position of the node on 3D mesh 10. Process 28 may assign the node to a polygon in the 3D mesh that is closest to the original position. Process 28 may assign the node to the candidate that is closer to the destination node.

Alternatively, process 28 may re-locate (409) the node as follows. Process 28 may obtain a current position of the path finding process on 3D mesh 40 and assign the predetermined node in accordance with the current position. For example, if process 28 determines that a node is missing because it was not encountered at an expected position, process 28 determines its current location on the node graph and assigns the missing node to a polygon nearest the current location. This technique is particularly applicable if the change in performance is detected during the path finding process.

If, on the other hand, none of the predetermined nodes has been deleted, process 28 simply re-performs the path finding process on the adjusted node graph, i.e., 3D mesh 40, in order to obtain node path 42.

Once the node path has been determined, process 28 moves (411) a 3D model along the node path.

Fig. 7 shows a computer 46 for determining a node path and for rendering 3D models using process 28. Computer 46 includes a processor 48, a storage medium 50 (e.g., a hard disk), and a 3D graphics accelerator 52 for processing 3D data (see view 54). Storage medium 50 stores 3D data 56 that defines an animation sequence that includes 3D mesh 10, and machine-executable instructions 58 for performing process 28. Processor 48 and/or graphics accelerator 52 execute instructions 58 to perform process 28 on 3D mesh 10.

Process 28, however, is not limited to use with any particular hardware or software configuration; it may find applicability in any computing or processing environment. Process 28 may be implemented in hardware, software, or a combination of the two. Process 28 may be implemented in one or more computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory

and/or storage elements), at least one input device, and one or more output devices. Program code may be applied to data entered using an input device to perform process 28 and to generate output information. The output information may be applied to one or more output devices.

Each such program may be implemented in a high level procedural or object-oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language. The language may be a compiled or an interpreted language.

Each computer program may be stored on an article of manufacture, e.g., a storage medium, such as a CD-ROM, hard disk, or magnetic diskette, that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform process 28. Process 28 may also be implemented as a computer-readable storage medium, configured with a computer program, where, upon execution, instructions in the computer program cause the computer to operate in accordance with process 28.

Process 28 is not limited to the embodiments described herein. The blocks of process 28 are not limited to the order shown. Blocks in the run-time stage may also be performed in

the pre-processing stage and vice versa. Process 28 can be used with any type of 3D mesh, not just surface terrains. For example, process 28 may be used to determine a node path through a maze in a video game. Metrics other than those described above may be used.

Other embodiments not described herein are also within the scope of the following claims.

What is claimed is: